Spectrum™ Technology Platform

Version 8.0.0 SP3

Spectrum Spatial Administration Guide

Pitney Bowes Software

© 2013 Pitney Bowes Software Inc. All rights reserved. MapInfo and Group 1 Software are trademarks of Pitney Bowes Software Inc. All other marks and trademarks are property of their respective holders.

USPS® Notices

Pitney Bowes Inc. holds a non-exclusive license to publish and sell ZIP + 4® databases on optical and magnetic media. The following trademarks are owned by the United States Postal Service: CASS, CASS Certified, DPV, eLOT, FASTforward, First-Class Mail, Intelligent Mail, LACS^{Link}, NCOA^{Link}, PAVE, PLANET Code, Postal Service, POSTNET, Post Office, RDI, Suite^{Link} , United States Postal Service, Standard Mail, United States Post Office, USPS, ZIP Code, and ZIP + 4. This list is not exhaustive of the trademarks belonging to the Postal Service.

Pitney Bowes Inc. is a non-exclusive licensee of USPS® for NCOA^{Link}® processing.

Prices for Pitney Bowes Software's products, options, and services are not established, controlled, or approved by USPS® or United States Government. When utilizing RDI™ data to determine parcel-shipping costs, the business decision on which parcel delivery company to use is not made by the USPS® or United States Government.

Centrus Notices

Centrus Data Products contained on this media and used within Centrus applications are protected by various trademarks and by one or more of the following copyrights:

© Copyright United States Postal Service. All rights reserved.

© 2011 TomTom. All rights reserved. TomTom and the TomTom logo are registered trademarks of TomTom N.V.

© Copyright NAVTEQ. All rights reserved

© Copyright United States Census Bureau

© Copyright Nova Marketing Group, Inc.

Portions of this program are © Copyright 1993-2007 by Nova Marketing Group Inc. All Rights Reserved

© Copyright Canada Post Corporation

This CD-ROM contains data from a compilation in which Canada Post Corporation is the copyright owner.

© 2007 Claritas, Inc.

ICU Notices

Copyright © 1995-2011 International Business Machines Corporation and others.

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

GeoNames Notices

The World Geocoding data set contains data licensed from the GeoNames Project (**www.geonames.org**) provided under the Creative Commons Attribution License ("Attribution License") located at **http://creativecommons.org/licenses/by/3.0/legalcode**. Your use of the GeoNames data (described in the Spectrum™ Technology Platform User Manual) is governed by the terms of the Attribution License, and any conflict between your agreement with Pitney Bowes Software, Inc. and the Attribution License will be resolved in favor of the Attribution License solely as it relates to your use of the GeoNames data.

Pitney Bowes Software
Documentation Team
pbbidocs@pb.com

July 18, 2013

Contents

Chapter 1: Configuring Your System....9

Changing the Default Port Number for Spectrum Spatial....10

Changing Your Repository Database....11

Set Up a PostgreSQL Repository Database 11

Set Up an Oracle Database 13

Accessing the Repository using WebDAV....14

Reload the Service Configuration using JMX Console....15

Uploading and Accessing Resources using Third Party Tools....15

Using WebFolders to Access the Repository Resources....15

Using DAVExplorer to Access the Repository Resources....16

Configuring the Web Services....17

About Web Service Configurations....18

How to Change Web Service Configuration Settings....18

Running Spectrum™ Technology Platform as a Linux Service....18

Steps....19

Chapter 2: Managing Security....21

Introduction....22

What Is the User Management Service?....22

Service URL Formats....22

Creating and Managing Users For Spectrum™ Technology Platform....22

Rules Using the User Management Service SOAP Interface....23

Managing Users....23

Starting the Management Console....23

Enabling User Permissions For Consoles....24

Managing User Accounts....24

Turning off Security for Services and the Repository....25

Setting User Permissions....27

GetPermissionsRequest....27

SetPermissionsRequest....28
AddPermissionsRequest....29
RemovePermissionsRequest....30

Chapter 3: Monitoring Your System....33
Event Log....34
Viewing the Event Log....34
Setting Event Log Options....34
Spatial Logging....35
E-mail Notification....37
Configuring License Expiration Notification....37
Viewing Version Information....38
Viewing and Exporting License Information....38
Using the JMX Console Performance Monitor....38
Monitoring Memory Usage....39

Chapter 4: Managing Memory and Threading....41
Introduction to Managing Memory and Threading....42
Spectrum Performance Tuning....42
JVM Tuning....42
Remote Component Configuration....42
Tuning the Components Using MaxActive....42
Increasing Heap Memory for Spatial Components....43
Increasing Heap Memory for the Platform....44

Chapter 5: Load Balancing Spatial Services Tutorial....45
About this Tutorial....46
Deployment Architecture....46
Install Spectrum....48
Set Up a Map Image File Share....49
Configure Spectrum....49
Add the Map File Share to Spectrum....49
Modify the Service Configurations....50
Modify Java Properties File....50
Configure Ports for Multiple Spectrum Instances....50
Configure Common Repository....50
Bulk Export Using WebDAV....51

Set Up the Common Repository Database 51
Import the Repository Content.....53
Shared Spectrum Local Data.....54
Performance Tuning.....54
Set Up Load Balancer.....54

Chapter 6: Troubleshooting Your System.....57
Rebuilding a Corrupt Repository Index.....58
Monitoring Memory Usage of a Non-Responsive Server.....58

1

Configuring Your System

In this section:

- Changing the Default Port Number for Spectrum Spatial .10
- Changing Your Repository Database 11
- Accessing the Repository using WebDAV 14
- Uploading and Accessing Resources using Third Party Tools 15
- Configuring the Web Services 17
- Running Spectrum™ Technology Platform as a Linux Service 18

Changing the Default Port Number for Spectrum Spatial

After you install Spectrum, you can change the default port settings that were assigned during installation by manually editing the global, startup, and individual service configuration files (particularly the port of the repository). There are several reasons for needing to change the default port number:

- Currently, the silent installer for Spectrum does not allow you to specify the port; it can only be specified after the install.
- A port conflict occurs after the install.
- You need a proxy on port 8080 but have a limited number of ports to expose externally, so you would like to move Spectrum without re-creating all your settings and data flows.
- You want to try out a new version of Spectrum without removing your old one. Since you cannot install them both, you can turn off the existing one and put down a Spectrum image which uses a different port.

Note: This task is only for experienced administrators who have application server experience changing port numbers, as network port conflicts can result in module components failing to start up. One indication that a component has failed to start up is if it does not appear in the Management Console. To troubleshoot the problem, look at the Spectrum Spatial server log file. This log shows which port is causing the problem. You can find the Spectrum Spatial Server log file in: [install folder]\server\app\repository\logs\server.log. The install folder default is C:\Program Files\Pitney Bowes\Spectrum.

To change the default port number, you can either copy the entire configuration folder then edit the files locally, or you can edit the configuration files in place. If you copy locally, you cannot put the configuration folder back while the server is running; you must restart all the Spatial services after making the changes. If you edit in place, you do not need to stop the server; however, the changes will not take effect until you restart the server.

The following network ports are used by default:

- 1701—Internal database connection
- 8080—HTTP communication
- 10119—Proprietary SOCKET client communication protocol
- 10389—LDAP databases

To change the default port number, with Spectrum Spatial running:

1. Decide whether to edit a local copy of the configuration files or in place. If you are going to edit the configuration files locally, use WebDAV to copy the Configuration folder from the repository to a local disk.
2. Edit all the configuration files (either in place or in a local copy) by changing all the ports.

   Note: There are other ports besides the repository port (for example, where the map images get accessed, the access port for WMS, WFS, and CSW). Change all references to the new port.

3. Stop the Spectrum server. (via the tray control or services.msc).
4. Update the spectrum-container.properties file in [install folder]\server\app\conf (for example, C:\Program Files\Pitney Bowes\Spectrum\server\app\conf):

   ```
   # Server ports
   spectrum.http.port=8080
   ```

5. Update the java.properties file for Spatial in [install folder]\server\modules\spatial (for example, C:\Program Files\Pitney Bowes\Spectrum\server\modules\spatial. Change all references to port numbers for each service.
6. Restart Spectrum.

   - If you edited the configuration files in place everything should be working.
   - If you edited locally, Spectrum should be working but not much of Spectrum Spatial will be working, since all the repository URL ports have the old values.

   1. Copy in the entire Configuration folder or its files from local copy to the Configuration folder of the repository via WebDAV.
   2. Restart the Spatial services via JMX (one by one) or restart the server.

Changing Your Repository Database

Spectrum stores named resources (maps, layers, tables and styles), geographic metadata and configuration in a repository. In the default single server installation an embedded database is used to store these resources on the local server. There are several reasons you may need to use a database other than the embedded Derby database:

- To create a scalable solution that uses a resilient independent database.
- To use an in-house database preferred or dictated by your company.

In this release, Spectrum supports Oracle, PostGreSQL(PostGIS) and Microsoft SQL Server as repository databases.

Set Up a PostgreSQL Repository Database

These steps describe how to set up your repository on a PostgreSQL database:

1. Copy all resources to a local folder using WebDAV.
2. Back up the folder /<spectrum root>/server/modules/spatial/jackrabbit to a local directory or disk.
3. Stop Spectrum.
4. Add the database JDBC drivers to the Spectrum common lib directory to allow it to use the selected database.

   Copy the /<spectrum root>/server/modules/spatial/lib/postgresql-8.4-701.jdbc4.jar file to /<spectrum root>/server/app/lib/postgresql-8.4-701.jdbc4.jar.

5. Edit the /<spectrum root>/server/modules/spatial/jackrabbit/repository.xml file to point the repository to a database and add clustering. There are four separate changes you need to make:

a) Modify the two FileSystem sections within the Repository and Workspace sections of the file:

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
   <param name="driver" value="org.postgresql.Driver"/>
   <param name="url"
value="jdbc:postgresql://<hostname>:<port>/<databasename>"/>
   <param name="schema" value="postgresql"/>
   <param name="user" value="<user>"/>
   <param name="password" value="<pwd>"/>
   <param name="schemaObjectPrefix" value="rep_"/>
  </FileSystem>
```

b) Modify the Persistence Manager within the Workspace:

```
<PersistenceManager
class="org.apache.jackrabbit.core.persistence.bundle.PostgreSQLPersistenceManager">

   <param name="url"
value="jdbc:postgresql://<hostname>:<port>/<databasename>"/>
   <param name="schema" value="postgresql"/>
   <param name="user" value="<user>"/>
   <param name="password" value="<pwd>"/>
   <param name="schemaObjectPrefix" value="${wsp.name}_"/>
   <param name="externalBLOBs" value="false"/>
  </PersistenceManager>
```

c) Enable Clustering at the end of the file, right above the </Repository> tag. Each instance of Spectrum will need to have a distinct id to enable synchronization of clustering to work. The delay defines the time delay for synchronization in milliseconds.

```
<Cluster id="node1" syncDelay="2000">
   <Journal class="org.apache.jackrabbit.core.journal.DatabaseJournal">

    <param name="revision" value="${rep.home}/revision.log" />
    <param name="driver" value="org.postgresql.Driver" />
    <param name="url"
value="jdbc:postgresql://<hostname>:<port>/<databasename>" />
    <param name="schema" value="postgresql"/>
    <param name="schemaObjectPrefix" value="rep_"/>
    <param name="user" value="<user>"/>
    <param name="password" value="<pwd>"/>
    <param name="databaseType" value="postgresql"/>
   </Journal>
  </Cluster>
```

d) Comment out the DataStore section:

```
<DataStore class="org.apache.jackrabbit.core.data.FileDataStore"/>
```

6. Restore the resources by copying them from the local folder into the Repository using WebDAV.
7. Remove the following folders from the /server/modules/spatial/jackrabbit directory for each instance of Spectrum: repository, version, workspaces.
8. If your PostgreSQL database has previously had repository content added, you must remove tables from your database so a clean repository can be created. If you are starting with a new database, please make sure the tables do not exist. The following tables need to be removed from the database:

```
public.default_names_id_seq
public.default_binval
public.default_bundle
public.default_names
public.default_refs
```

```
public rep_fsentry
public.rep_global_revision
public.rep_journal
public.rep_local_revisions
public.security_binval
public.security_bundle
public.security_names
public.security_refs
```

Set Up an Oracle Database

These steps describe how to set up your repository on an Oracle database:

1. Copy all resources to a local folder using WebDAV.
2. Back up the folder /<spectrum root>/server/modules/spatial/jackrabbit to a local directory or disk.
3. Stop Spectrum.
4. Add the database JDBC drivers to the Spectrum common lib directory to allow it to use the selected database.

   Copy the /<spectrum root>/server/modules/spatial/lib/postgresql-8.4-701.jdbc4.jar file to /<spectrum root>/server/app/lib/postgresql-8.4-701.jdbc4.jar.
5. Edit the /<spectrum root>/server/modules/spatial/jackrabbit/repository.xml file to point the repository to a database and add clustering. There are four separate changes you need to make:
   a) Modify the two FileSystem sections within the Repository and Workspace sections of the file:

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
            <param name="driver" value="oracle.jdbc.OracleDriver" />
            <param name="url"
value="jdbc:oracle:thin:@//<hostname>:<port>/<databasename>" />
            <param name="user" value="<user>" />
            <param name="password" value="<pwd>" />
            <param name="schema" value="oracle"/>
            <param name="schemaObjectPrefix" value="rep_"/>
            <param name="tableSpace" value=""/>
</FileSystem>
```

   b) Modify the Persistence Manager within the Workspace:

```
<PersistenceManager
class="org.apache.jackrabbit.core.persistence.pool.OraclePersistenceManager">

            <param name="driver" value="oracle.jdbc.OracleDriver" />
            <param name="url"
value="jdbc:oracle:thin:@//<hostname>:<port>/<databasename>" />
            <param name="user" value="<user>" />
            <param name="password" value="<pwd>" />
            <param name="schema" value="oracle"/>
            <param name="schemaObjectPrefix" value="${wsp.name}_"/>
            <param name="tableSpace" value=""/>
            <param name="externalBLOBs" value="false"/>
        </PersistenceManager>
```

c) Enable Clustering at the end of the file, right above the </Repository> tag. Each instance of Spectrum will need to have a distinct id to enable synchronization of clustering to work. The delay defines the time delay for synchronization in milliseconds.

```
<Cluster id="node1" syncDelay="2000">
        <Journal
class="org.apache.jackrabbit.core.journal.OracleDatabaseJournal">
            <param name="driver" value="oracle.jdbc.OracleDriver" />
            <param name="url"
value="jdbc:oracle:thin:@//<hostname>:<port>/<databasename>" />
            <param name="user" value="<user>" />
            <param name="password" value="<pwd>" />
            <param name="databaseType" value="oracle"/>
            <param name="revision" value="${rep.home}/revision.log" />
        </Journal>
    </Cluster>
```

d) Comment out the DataStore section:

```
<DataStore class="org.apache.jackrabbit.core.data.FileDataStore"/>
```

6. Restore the resources by copying them from the local folder into the Repository using WebDAV.
7. Remove the following folders from the /server/modules/spatial/jackrabbit directory for each instance of Spectrum: repository, version, workspaces.
8. If your Oracle database has previously had repository content added, you must remove tables from your database so a clean repository can be created. If you are starting with a new database, please make sure the tables do not exist. The following tables need to be removed from the database:

```
public.default_names_id_seq
public.default_binval
public.default_bundle
public.default_names
public.default_refs
public rep_fsentry
public.rep_global_revision
public.rep_journal
public.rep_local_revisions
public.security_binval
public.security_bundle
public.security_names
public.security_refs
```

Accessing the Repository using WebDAV

Configuration files are pre-loaded in the repository for each service. These configuration files are located at **http://localhost:8080/RepositoryService/repository/default/Configuration/**.

To configure the services, you must use a WebDAV protocol tool to access the JCR repository, copy the configuration file locally, make changes to the file, copy the configuration file back to the repository, and reload the configuration using the JMX Console. There are many tools available to accomplish the WebDAV connection tasks. We have provided examples using **WebFolders** and **DAVExplorer**.

Reload the Service Configuration using JMX Console

Once you have modified a service configuration, you must reload the configuration in the repository using the JMX Console. The JMX console allows you to reload and administer a service, without having to restart the application container.

To reload the service configuration:

1. Access the JMX Console using the following URL: **http://localhost:8080/jmx-console/**
2. Under the `Domain: EnterpriseMapping` section, select the administration link for the service. For example, `EnterpriseMapping:name=Administration,type=WMS Service`.
3. Click the `Invoke` button for the `reloadConfiguration` operation.

You will get a message on the status of the invocation.

Uploading and Accessing Resources using Third Party Tools

Named resource files are stored in the repository. A number of sample files that ship with Spectrum™ Technology Platform are located at **http://localhost:8080/RepositoryService/repository/default/Samples** under a particular folder. For example:

- NamedLayers
- NamedMaps
- NamedStyles
- NamedTables
- NamedTiles

For your own named resources, you can create any folder name you wish.

You can access these files manually using a WebDAV compliant tool. This section describes the manual method. To access resources manually, you must use a WebDAV protocol tool to access the JCR repository. There are many tools available to add and access resources in the repository using the WebDAV. We have provided two examples:

Using WebFolders to Access the Repository Resources

To add or modify a resource, you must copy the resource to or from the repository using a WebDAV tool. Using WebFolders is an easy way to access the repository and the resources contained in the repository.

Note: WebFolders is for Windows machines only. To access the repository, you must be on the same machine where Spectrum™ Technology Platform and the repository are installed.

To configure a WebFolder:

1. Using the Windows file explore, for the method appropriate for your version of Windows, select `Map Network Drive...`
2. In the pop-up window, click on the link '`sign up for online storage or connect to a network server`' or you may have a link that mentions'`Connect to a website...`'.
3. Click the `Next` and select `Choose another network location`. Click `Next`.
4. In the `Internet or network address`field add the repository URL `http://localhost:8080/RepositoryService/repository/default/`. Click `Next`.
5. If you are asked for credentials, use the username and password `Admin/Admin`
6. Give this connection a name. For example, `Spatial Server Repository`. Click `Next`.

Once finished, you will have a folder connection to the contents of the repository under your network places.

The WebFolder connection to the repository can be used like any other windows explorer folder, where files can be copied from the repository, modified, and copied back to the same location in the repository.

Using DAVExplorer to Access the Repository Resources

To add or modify a resource, you must copy the resource to or from the repository using a WebDAV tool. Using DAVExplorer is an easy way to access the repository and the resources contained in the repository. DAVExplorer is a freely available WebDAV client application. This software is available from **http://www.davexplorer.org**.

Note: DAVExplorer is for Windows machines only. To access the repository, you must be on the same machine where Spectrum™ Technology Platform and the repository are installed.

To get or add resources from the repository using DAVExplorer, use the following instructions:

Getting Resources From the Repository Using DAVExplorer

Use the following steps to get resources from the repository using DAVExplorer:

1. Open DAVExplorer.
2. In DAVExplorer, enter the URL of the Spectrum™ Technology Platform repository and click the **Connect** button.

   For example, enter `localhost:8080/RepositoryService/repository/default/`. (Note that DAVExplorer prepends `http://` automatically.)

   If prompted, enter the `admin/admin` login name and password required to connect to the repository.

   Once you are connected to the repository, a node for the repository appears in the treeview pane on the left.
3. In the treeview pane on the left, expand the nodes under the repository node until you see the node that contains the type of resource you want to get.

   For example, if the named resource you want to get is a configuration, expand the repository nodes until you see the **Configuration** node. Click on the node to select it. The named configuration resources in the repository are then listed in the right pane.
4. In the right pane, click on the resource you want to get.

   You may click on any of the fields of the named resource to select it.
5. On the **File** menu, select **Get File**.

The **Save As** dialog box opens.

6. In the **Save As** dialog box, enter a name for the named resource definition file and select the directory in which you want to save it, then click the **Save** button.

   The selected named resource definition file is saved to the selected file location.

   Note: You should always save the resource as the same name as it appears in the repository. By using this technique, you will never have a conflict when adding the resource back to the repository.

Adding Resources to the Repository Using DAVExplorer

Use the following steps to add resources to the repository using DAVExplorer:

1. Open DAVExplorer.
2. In DAVExplorer, enter the URL of the Spectrum™ Technology Platform repository and click the **Connect** button.

   For example, enter `localhost:8080/RepositoryService/repository/default/`. (Note that DAVExplorer prepends `http://` automatically.)

   If prompted, enter the `admin/admin` login name and password required to connect to the repository.

   Once you are connected to the repository, a node for the repository appears in the treeview pane on the left.

3. In the treeview pane on the left, expand the nodes under the repository node until you see the node that corresponds to the type of resource you are adding.

   For example, if you are adding a configuration resource, expand the repository nodes until you see the **Configuration** node. Click on the node to select it.

4. On the **File** menu, select **Write File**.

   The **Write File** dialog box opens.

5. In the **Write File** dialog box, select the definition file of the resource you want to add to the repository, then click the **Open** button.

   The selected resource is added to the repository.

Configuring the Web Services

This section provides information about how to configure the Location Intelligence Module web services.

About Web Service Configurations

You can, and frequently must, explicitly specify the desired behavior of the Location Intelligence Module web services via settings in each web service's configuration file. The configuration file for each web service is held in the Location Intelligence Module repository as a named configuration.[1]

Note: Named configurations are not like other named resources that are held in the repository. You cannot use the Named Resource Service to access named configurations. Instead, you must use a WebDAV tool of your choice, such as DAVExplorer or Windows web folders.

For information about the name and location of each web service's named configuration in the repository, as well as a list of the configuration parameters for each web service, refer to the "Working With Spatial Services" chapter in the *Spectrum Spatial Developer Guide*.

How to Change Web Service Configuration Settings

To change web service configuration settings:

1. Pull the named configuration file for the web service out of the repository using your favorite WebDAV tool.

   Note: You cannot use the Named Resource Service to extract a named configuration file from the repository.

2. Using a text editor, make any required changes to the named configuration file.
3. Re-add the named configuration file back into the repository using your favorite WebDAV tool.

   Note: You cannot use the Named Resource Service to add a named configuration file to the repository.

4. Do one of the following to reload the web service configuration:

   - Restart the web service.
   - Use the Spectrum™ Technology Platform JMX Console (available at `http://hostname[:portnumber]/jmx-console`) to reload the configuration without restarting the web service.

Running Spectrum™ Technology Platform as a Linux Service

This tutorial will show you the steps you need to follow to run Spectrum™ Technology Platform as a Linux service.

[1] The Geometry Service alone does not have a corresponding named configuration because the Geometry Service has no configurable settings.

Steps

These instructions show how to run the Spectrum™ Technology Platform as a Linux service. In addition to these instructions, a script is available for download. You can obtain the script here: **http://reference.mapinfo.com/software/spectrum/lim/8_0/downloads/linux_script.tgz**

To run as a Linux service:

1. Modify the provided `pbspectrum` script.
   a) Modify the `chkconfig` parameter at line# 5. By Default this parameter is: `# chkconfig: 35 90 10`

      First value(35) is runlevel. Use 'man init' for more information.

      Second value(90) is start priority

      Third value(10) is stop priority.

      Start and stop priority should be set according to the dependent services. For example, if Oracle Server is running on the same machine and is used by Spectrum™ Technology Platform then the Spectrum™ Technology Platform starting priority should be less than the Oracle Service and stopping priority should be higher than the Oracle service. Use 'man chkconfig' for more information.

   b) Modify SPECTRUM_ROOT variable at line #11 with your Spectrum™ Technology Platform installation directory.
2. Copy the modified `pbspectrum` script to either `/etc/rc.d/init.d` for RedHat Linux or `/etc/init.d` for Suse Linux.
3. Change the mode of the `pbspectrum` script to executable. `/etc/rc.d/init.d` for RedHat Linux or `/etc/init.d` for Suse Linux.

   `cd /etc/init.d` or `cd /etc/rc.d/init.d` depending on your Linux version.

   run `chmod +x pbspectrum`
4. Run `chkconfig --add pbspectrum`
5. Verify the script is working by restarting the machine. Use `shutdown -r now` to reboot from shell.

Once completed, you may also use the following:

- `service pbspectrum start` to start Spatial Server
- `service pbspectrum stop` to stop Spatial Server
- `service pbspectrum restart` to restart Spatial Server

Note: The provided script runs the command 'ulimit -n 8192' which is required to increase the number of open files in Linux.

2

Managing Security

In this section:

- Introduction 22
- Managing Users 23
- Setting User Permissions 27

Introduction

This chapter provides a basic introduction to the User Management Service. It describes what the User Management Service is and rules for using it.

What Is the User Management Service?

The User Management Service provides a simplified interface to manage security for the repository, focused on how to restrict who can access the resources in the repository. Setting security allows you to expose or restrict different resources (subsets of your data and resources) to different users or departments. To enforce this, security has been added to Spectrum™ Technology Platform that allows you to specify which users get to see what resources.

The Spectrum™ Technology Platform repository security is managed using an internal ACL (Access Control List). This allows you to specify which users are granted access to resources, as well as what operations are allowed on given resources. The operations for repository user management are performed using the User Management SOAP interface.

Service URL Formats

The URL endpoint for the User Management SOAP service has the following general form:

```
http://localhost:8080/soap/UserManagementService
```

The URL for the User Management WSDL has the following general form:

```
http://localhost:8080/soap/UserManagementService?wsdl
```

The URL for the User Management service Demo page has the following general form:

```
http://localhost:8080/Spatial/UserManagementService/DemoPage.html
```

Creating and Managing Users For Spectrum™ Technology Platform

Creating and managing users is a two step process:

1. Create the user using the Spectrum™ Technology Platform Management Console. This allows the user to authenticate with the Spectrum™ Technology Platform services.
2. Give the user permissions using the User Management Service SOAP interface. This allows the user to access resources in the repository.

Note: You do not have to add the admin or guest users to Spectrum™ Technology Platform. These users have already been created.

Rules Using the User Management Service SOAP Interface

The following rules apply when setting permissions for users using the User Management SOAP Interface:

1. You must first have created users in the Spectrum™ Technology Platform Management Console (giving them access to the services).
2. There is a default 'everyone' user group that is applied to repository nodes and resources when you do not specify set permissions. This user group has READ permissions. So all users have READ permissions on a repository node or resource unless modified using the User Management SOAP Interface.
3. It is preferred that you set permissions on a repository node (folder) rather than a specific resource. This makes repository management easier to maintain.
4. You need to provide a user `read`, `add`, and `modify` permissions to allow them the ability to add Named Tables using the Management Console, modify any resources in the repository, add or modify any resources using the Named Resource Service, or perform a harvest operation using the CSW Service.
5. You do not have to add the admin or guest users. These users have already been created.

The following permissions are required for performing the following actions, either directly using WebDAV or WebFolder, using the Resource Management service, or harvesting metadata using the CSW service:

Action	Read	Add	Remove	Modify	All
Access a subfolder	X				
Add a subfolder	X	X			
Remove a subfolder	X		X		
Add files to a folder	X	X		X	
Remove files from a folder	X		X	X	
Update files in a folder	X			X	
Modify permissions of a folder					X

Managing Users

This section describes how to manage users in Spectrum™ Technology Platform, specifically create, modify, and delete users that access the management consoles and services. You must create all users using the Management Console, and set permissions to access the repository using the User Management Service.

Starting the Management Console

Start the Spectrum™ Technology Platform Management Console by selecting **Start** > **Programs** > **Pitney Bowes** > **Spectrum™ Technology Platform** > **Client Tools** > **Management Console** from your desktop.

To connect to the Spectrum™ Technology Platform Management Console:

1. Type in the server name or select it from the drop-down list.

   Note: If you have multiple instances of the Management Console accessing the same Spectrum™ Technology Platform server, it is possible for one user to overwrite another user's changes. Therefore, it is recommended that you do not run multiple instances of the Management Console against the same server.

2. Enter your user name, password, and the port number.
3. Click the Use secure connection box if you want communication between the client and the server to take place over an HTTPS connection.
4. Click **Login**. If this is your first time connecting, the user will default to "guest" and you can connect with no password.

   Note: The default port number is 8080 for HTTP connections. Use the port number appropriate for your environment. Once you have successfully connected, this value will default for the next connection attempt.

Enabling User Permissions For Consoles

Spectrum™ Technology Platform can enforce permissions on user accounts when accessing the client consoles, providing you with additional control of the actions a user can take.

1. Open the Management Console.
2. Expand **Security** then click **Options**.
3. Click the Limit access according to user permissions checkbox to limit access to the permissions established for individual users.

   For instructions on defining permissions for each user, see **Adding a New User** on page 24, **Modifying a User** on page 25, or **Deleting a User** on page 25.

Managing User Accounts

This section describes you how to create users and set user security privileges for Spectrum™ Technology Platform consoles.

Adding a New User

1. Open the Management Console.
2. Expand **Security** then click **Users**.
3. Click **Add**. The **New User** window appears.
4. Enter the user's name in the **User name** field.

   Note: You do not have to add the admin or guest users to the Management Console. These users have already been created.

5. Enter the user's password in the **Password** field.
6. Confirm the user's password in the **Confirm password** field.

7. Enter a description of the user in the **Description** field.
8. Check **Administrative Privileges** if you would like the user to have the rights needed to run the Management Console, Enterprise Designer, and Interactive Driver. If the user does not have admin rights, they can still access services on the system. Click **OK**.

Modifying a User

1. Open the Management Console.
2. Expand **Security** then click **Users**.
3. Select the user whose permissions you want to modify and click **Modify**. The **User Properties** window appears in which can modify the user name, password, description, and administrative access. Note that the length of the password is masked by the display of 24 asterisks.
4. Click **OK** to save your changes.

Deleting a User

1. Open the Management Console.
2. Expand **Security** then click **Users**.
3. From the **User Management** screen, select the user you want to delete and click **Delete**.
4. Click **Yes** to delete or **No** to cancel.

   Note: Admin and guest accounts cannot be deleted.

Turning off Security for Services and the Repository

All services and access to resources used by the Spectrum™ Technology Platform Location Intelligence Module are configured, by default, with authentication turned on. This allows certain functionality to restrict access to resources and ability to modify resource in the repository. For example, the Named Resource Service AddResource operation, and the CSW service Harvest operation both require authentication, as both of these operations require write permissions to the repository.

You can turn off permissions to all or any combination of the following:

- resources in the repository
- REST services
- SOAP services

Turning off authentication for all services and the repository is useful if you have your own high-level authentication built into the solution that is using the Location Intelligence Module services.

Note: If you turn off authentication for one or both of the services, you should still pass user credentials so the repository is still managing permissions on resources, otherwise by default the admin user is used (who has full rights), and all resources are accessible in the repository.

Turning off Security for the Repository

To turn off repository security:

1. Launch the User Management Service Demo page at **http://localhost:8080/Spatial/UserManagementService/DemoPage.html** (replacing localhost and port 8080 with your correct configuration).
2. Using `admin` credentials in the User and Password fields, set the `everyone` user with the `all` permission using the following request:

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.mapinfo.com/midev/service/usermanagement/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:SetPermissionsRequest>
         <v1:UserName>everyone</v1:UserName>
         <v1:ResourcePath>/</v1:ResourcePath>
         <v1:Permissions>
            <v1:Permission>all</v1:Permission>
         </v1:Permissions>
         <v1:Recursive>false</v1:Recursive>
      </v1:SetPermissionsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Once finished, security is turned off for the repository.

Turning off Security for the REST Services

To turn off security for REST services:

1. Access the JMX Console using the following URL: **http://localhost:8080/jmx-console/** (replacing localhost and port 8080 with your correct configuration).
2. Under the `Domain: com.pb.spectrum.platform.config` section, select the administration link for the `WebServiceSecurityConfigurationManager`.
3. For `RestServiceSecurityType` enter `OPEN` in the value field and click `set`.

Once finished, security is turned off for the REST services.

Turning off Security for the SOAP services

To turn off security for SOAP services:

1. Access the JMX Console using the following URL: **http://localhost:8080/jmx-console/** (replacing localhost and port 8080 with your correct configuration).
2. Under the `Domain: com.pb.spectrum.platform.config` section, select the administration link for the `WebServiceSecurityConfigurationManager`.
3. For `SoapServiceSecurityType` enter `OPEN` in the value field and click `set`.

Once finished, security is turned off for the SOAP services.

Setting User Permissions

This section introduces the User Management SOAP Interface for managing users and permissions for resources in the repository. This interface allows you to get, set, add, or remove permissions for a user.

Use the demo page for the User Management Service as a quick tool for managing user permissions. Simply modify the sample requests to meet your needs. The User Management Service demo page is located at **http://localhost:8080/Spatial/UserManagementService/DemoPage.html**

The User Management Service provides the following operations:

GetPermissionsRequest

Returns the permissions for a particular user for a specified repository node or resource.

Parameters

The following parameters are used:

Parameter	Example	Description
action	GetPermissionsRequest	Specifies the method name to get the permissions for a user.
UserName	user1	Specifies the user to return permissions.
ResourcePath	/NamedTables/WorldTable	Specifies the specific repository node (directory) or resource to return the permissions. The repository nodes or resources specified in resourcePath are listed from the top level of the repository http://localhost:8080/RepositoryService/repository/default/.

Example

The following example returns the permissions on the WorldTable resource for the user user1.

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.mapinfo.com/midev/service/usermanagement/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:GetPermissionsRequest>
         <v1:UserName>user1</v1:UserName>
         <v1:ResourcePath>/NamedTables/WorldTable</v1:ResourcePath>
      </v1:GetPermissionsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

SetPermissionsRequest

Defines the permissions for a particular user for a specified repository node or resource. When you set permissions, the basic read permissions are always kept for the user, however any additional permissions that were previously set or added are removed. For example if you set the `modify` permission for a user who currently had the `all` permission, that user will now have only `read` and `modify` permissions, and no longer have the `all` permission.

Parameters

The following parameters are used:

Parameter	Example	Description
`action`	`SetPermissionsRequest`	Specifies the method name to set permissions for a user.
`UserName`	`user1`	Specifies the user to set permissions.
`ResourcePath`	`/NamedTables/WorldTable`	Specifies the specific repository node (directory) or resource to set the permissions. The repository nodes or resources specified in resourcePath are listed from the top level of the repository `http://localhost:8080/RepositoryService/repository/default/`.
`Permissions`	`add`	Specifies the permissions. There are five valid permission types: `read`, `all`, `add`, `modify`, and `remove`.
`Recursive`	`false`	Specifies if this operation should be performed recursively on all child nodes of the given node in the repository. The default for recursive permission setting is false. If setting permissions on individual resources in the repository, the Recursive option will have no effect.

Example

The following example sets the permissions for user1 on the NamedTables node (and all child nodes) in the repository to add and modify. After performing this operation the user1 will have read, add, and modify permissions on the NamedTables node and all of the child nodes.

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.mapinfo.com/midev/service/usermanagement/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:SetPermissionsRequest>
         <v1:UserName>user1</v1:UserName>
```

```
         <v1:ResourcePath>/NamedTables</v1:ResourcePath>
         <v1:Permissions>
            <v1:Permission>add</v1:Permission>
            <v1:Permission>modify</v1:Permission>
         </v1:Permissions>
         <v1:Recursive>true</v1:Recursive>
      </v1:SetPermissionsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

AddPermissionsRequest

Adds new permissions to the users set of permissions for a specified repository node or resource. When you add permissions, the existing permissions are always kept for the user, and the new permissions are appended. For example if you add a modify permission for a user that currently has read and remove permissions, that user will now have read, remove, and modify permissions.

Parameters

The following parameters are used:

Parameter	Example	Description
`action`	`AddPermissionsRequest`	Specifies the method name to add permissions for a user.
`UserName`	`user1`	Specifies the user to add permissions.
`ResourcePath`	`/NamedTables`	Specifies the specific repository node (directory) or resource to add the permissions. The repository nodes or resources specified in resourcePath are listed from the top level of the repository `http://localhost:8080/RepositoryService/repository/default/`.
`Permissions`	`add`	Specifies the permissions. There are five valid permission types: `read`, `all`, `add`, `modify`, and `remove`.
`Recursive`	`false`	Specifies if this operation should be performed recursively on all child nodes of the given node in the repository. The default for recursive permission setting is false. If setting permissions on individual resources in the repository, the Recursive option will have no effect.

Example

The following example adds the modify permission for user1 on the NamedMaps node in the repository.

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.mapinfo.com/midev/service/usermanagement/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:AddPermissionsRequest>
         <v1:UserName>user1</v1:UserName>
         <v1:ResourcePath>/NamedMaps</v1:ResourcePath>
         <v1:Permissions>
            <v1:Permission>modify</v1:Permission>
         </v1:Permissions>
         <v1:Recursive>false</v1:Recursive>
      </v1:AddPermissionsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

RemovePermissionsRequest

Removes permissions from the users set of permissions for a specified repository node or resource. When you remove permissions, the specified permissions are removed from the existing set of permissions. This is the easiest way to restrict a user from accessing a particular resource. By removing the `read` permission for a user for a particular repository node or resource, they cannot be accessed by that user.

Parameters

The following parameters are used:

Parameter	Example	Description
`action`	`RemovePermissionsRequest`	Specifies the method name to remove permissions for a user.
`UserName`	`user1`	Specifies the user to remove permissions.
`ResourcePath`	`/NamedTables`	Specifies the specific repository node (directory) or resource to remove the permissions. The repository nodes or resources specified in resourcePath are listed from the top level of the repository `http://localhost:8080/RepositoryService/repository/default/`.
`Permissions`	`read`	Specifies the permissions. By removing the `read` permission, a user would no longer have access to a

Parameter	Example	Description
		resource. There are five valid permission types: `read`, `all`, `add`, `modify`, and `remove`.
`Recursive`	`false`	Specifies if this operation should be performed recursively on all child nodes of the given node in the repository. The default for recursive permission setting is false. If setting permissions on individual resources in the repository, the Recursive option will have no effect.

Example

The following example removes the read permission for user1 on the NamedTables node and all child nodes in the repository.

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.mapinfo.com/midev/service/usermanagement/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:RemovePermissionsRequest>
         <v1:UserName>user1</v1:UserName>
         <v1:ResourcePath>/NamedTables</v1:ResourcePath>
         <v1:Permissions>
            <v1:Permission>read</v1:Permission>
         </v1:Permissions>
         <v1:Recursive>true</v1:Recursive>
      </v1:RemovePermissionsRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

3

Monitoring Your System

In this section:

- Event Log 34
- Spatial Logging 35
- E-mail Notification 37
- Configuring License Expiration Notification 37
- Viewing Version Information 38
- Viewing and Exporting License Information 38
- Using the JMX Console Performance Monitor 38
- Monitoring Memory Usage 39

Event Log

Viewing the Event Log

1. Open the Management Console.
2. Expand **Event Log** then click **Events**. The window displays the information in the current log. These log entries include both system logging and service logging.
3. Click **Refresh** to view the latest entries.
4. Check **Show events upon open** to turn on (or turn off if already checked) the events log.

You can also view the event log by using a text editor and opening the file `<ServerLocation>\server\app\repository\logs\wrapper.log`.

Setting Event Log Options

You can specify the default logging level as well as logging levels for each service on your system. When you change logging levels the change will not be reflected in the log entries made before the change.

1. Open the Management Console.
2. Expand **Event Log** then click **Options**.
3. Click the **System default logging level** drop-down list to select an event logging level. Event logging levels include the following:
   - Disabled—no event logging enabled.
   - Fatal—minimal logging, logs only fatal errors. Fatal errors are those that make the system unusable.
   - Error—logs only errors and fatal errors. Errors make a single call unusable, possibly a single service, but not the whole system. The inability to load a specific service might be an error since other services would be available.
   - Warn—event warnings and errors are logged. Warnings indicate problems that do not stop the system from working (for example, when loading a service where a parameter has an invalid value, a warning is issued and the default parameter is used). During the use of a service, if results are returned but there is a problem, a warning will be logged. An example might be that casing was set to lower case, but Canadian does not support casing. Results are returned with a warning that the casing option was ignored.
   - Info—logging of high-level system information. This is the most detailed logging level suitable for production. Info level will typically be used during startup and initialization, providing product and version information, which services were loaded, etc.
   - Debug—a highly detailed level of logging, suitable for debugging problems with the system.
   - Trace—the most detailed level of logging, tracing program execution (method entry and exit). It provides detailed program flow information for debugging.

   Each logging level includes the ones above it on the list. In other words, if Warning is selected as the logging level, errors and fatal errors will also be logged. If Info is selected, informational messages, warnings, errors, and fatal errors will be logged.

   Note: Selecting the most intensive logging level can affect system performance. Therefore, you should select the least intensive setting that meets your particular logging requirements.

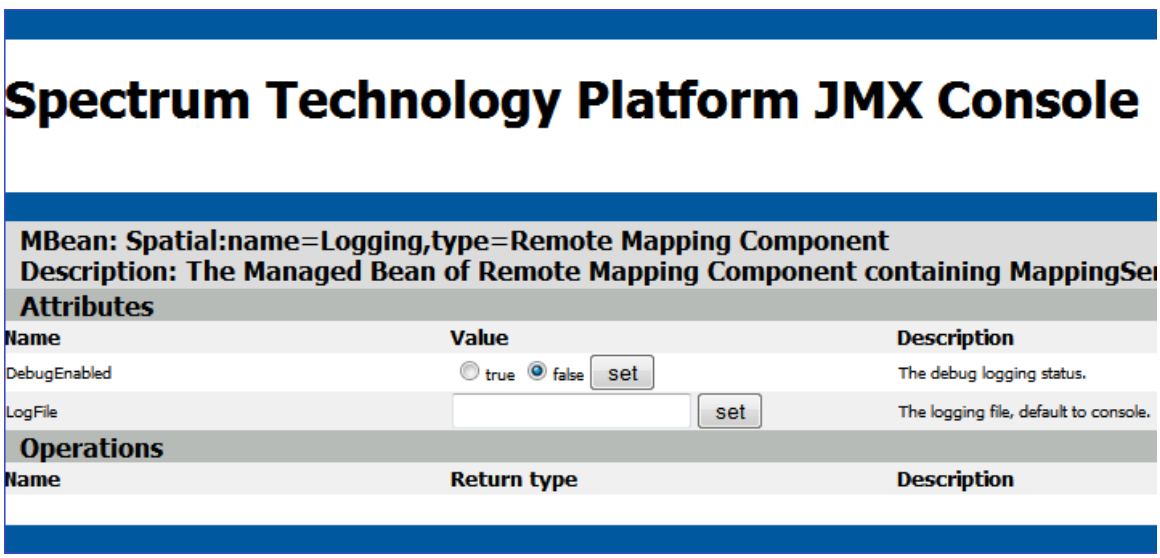
4. If you want to specify different logging levels for each service choose the logging level you want.

Spatial Logging

The JMX Console is equivalent to `log4j.category.com.mapinfo.midev=DEBUG,CONSOLE` in the logj4.properties file. If you want to see debug info for a short time, use JMX Console, otherwise, use logj4.

The remote components (feature and mapping) in the JMX Console can be configured individually:

- Spatial:name=Logging,type=Remote Feature Component
  - Feature Service
  - Geometry Service
  - Named Resource Service
  - User Management Service
  - WFS
  - CSW
- Spatial:name=Logging,type=Remote Mapping Component
  - Mapping Service
  - Map Tiling Service
  - WMS



Users can enable debugging and specify additional output file for each one of them. By default, log messages go into wrapper.log. Services in a component will output to the same file and cannot be further split. The configuration change made here will not persist, and will be lost after restart.

The log4j.properties file provides finer control on logging behavior, such as sending output to a log file instead of by default sending it to the console which redirects to the wrapper.log. You can also set the log level to turn off logging altogether or log only fatal errors, for example. As described above for the JMX Console, the log can only be redirected based on components (feature and mapping), not by services.

Default log4j file:

```
# Logger configuration for remote components

#Level: OFF, FATAL, ERROR, WARN, INFO, DEBUG
#Output: CONSOLE, LOGFILE

# log to console, redirected to Platform log
(server\app\repository\logs\wrapper.log)
# log to files, redirected to (server\modules\spatial\spatial.XXX.log)
log4j.category.com.mapinfo.midev=INFO,CONSOLE

# setup CONSOLE to append to the console
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=[${component.name}] %-5p %c
 %x - %m%n

# setup LOGFILE to append to the files
log4j.appender.LOGFILE=org.apache.log4j.RollingFileAppender
log4j.appender.LOGFILE.file=${g1.server.modules.dir}/spatial/${component.name}.log
log4j.appender.LOGFILE.maxFileSize=1024KB
log4j.appender.LOGFILE.maxBackupIndex=5
log4j.appender.LOGFILE.append=true
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=%d{ISO8601} %-5p %c %x - %m%n
# the end
```

Option	Values
Level	• OFF–turn off logging • FATAL–log severe errors that cause premature termination • ERROR–log runtime or unexpected errors • WARN–log warnings only; for example, using a deprecated API • INFO–log runtime events such as startup or shutdown [default] • DEBUG–log detailed debugging information
Output	• CONSOLE–sends log information to the JMX Console [default] • LOGFILE–sends log information to a log file based on component (feature or mapping)

E-mail Notification

Spectrum™ Technology Platform can alert you to potential problems to ensure that critical business processes are not interrupted. Notifications are sent as a result of conditions within dataflows and process flows. The messages can be formatted to contain context-sensitive information about the event that occurred.

1. Open the Management Console.
2. Expand **System** then click **Notification**.
3. On the **SMTP Settings** tab, enter a valid host name or IP address in the **Host** field.
4. Enter a valid port number or range in the **Port** field. The default is 25.
5. Enter the user name for logging on to the SMTP server in the **User Name** field.
6. Enter a password for logging on to the SMTP server in the **Password** field.
7. If you completed the **Password** field, re-enter the password for logging on to the SMTP server in the **Confirm Password** field.
8. Enter a valid e-mail address to where notification e-mail will be sent in the **From Address** field.
9. Enter a valid e-mail address to where notification e-mail will be sent in the **Test Address** field. This is used to ensure the notification process works.
10. Click **Test** to send a test message.

For information about the Expiration Settings tab, see **Configuring License Expiration Notification** on page 37.

Configuring License Expiration Notification

You can have Spectrum™ Technology Platform send an email notification when a license is about to expire.

1. Open the Management Console.
2. Expand **System** then click **Notification**.
3. Click the **SMTP Settings** tab. See **E-mail Notification** on page 37 for information about configuring the fields on this tab.
4. Click the **Expiration Settings** tab.
5. In the **Days before expiration to send notification** field, specify the number of days in advance that you want to be notified of a pending license or data expiration. For example, if you want to be notified 30 days before a license expires, specify `30`.
6. Check the **Send expiration notification** check box.
7. Click **Add** and specify the email address you want to receive the notification.
8. Select **File** > **Save**.

Viewing Version Information

1. Open the Management Console.
2. Expand **System** then click **Version Information**.
3. The Version Information window presents information on the configured services. Expanding the Server Information, System Information, Service Information, and Component Information folders will present the corresponding details. This includes versions numbers of the server, the operating system, the service software, and component versions.

   Note: This information is view-only.

Viewing and Exporting License Information

1. Open the Management Console.
2. Expand **System** then click **Licensing**.
3. Click the **Expiration Info** tab to view a list of licenses that are about to expire. Only licences that are within the timeframe specified on in the Notification node, Expiration Settings tab, are displayed.
4. Click the **License Information** tab to view a complete listing of all licenses installed on your system. To export your license information to a .lic file, click Export. This is helpful when resolving license issues with Technical Support.

Using the JMX Console Performance Monitor

The Spectrum™ Technology Platform JMX console provides a performance monitoring tool that records performance statistics for each stage in a dataflow. Use the JMX console to identify bottlenecks and observe the effects of different performance tuning adjustments.

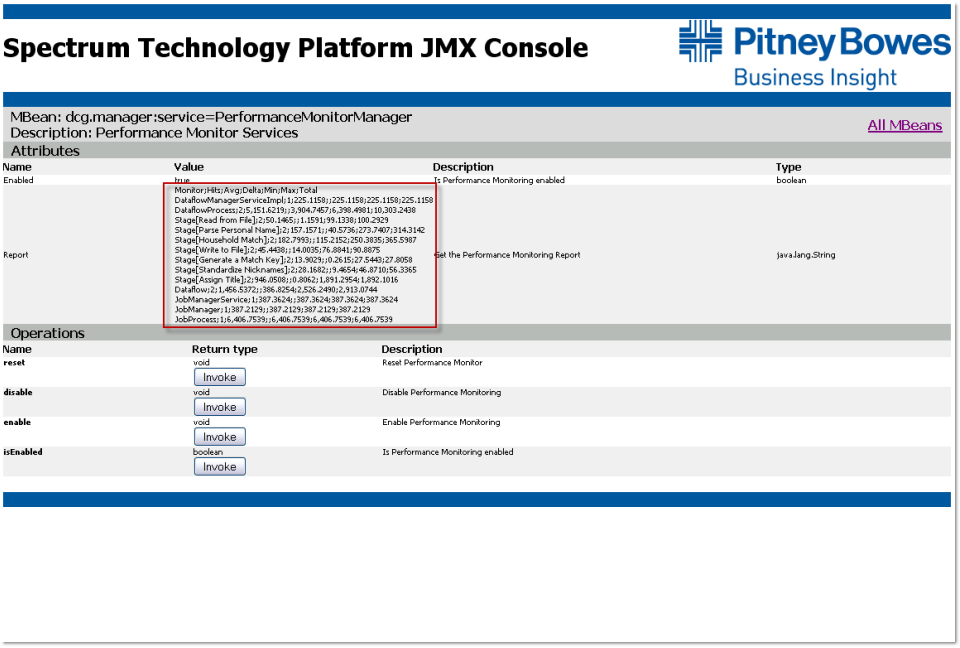
1. Open a web browser and go to `http://<server>:<port>/jmx-console`

   Where:

   <server> is the IP address or hostname of your Spectrum™ Technology Platform server.

   <port> is the HTTP port used by Spectrum™ Technology Platform. The default is 8080.
2. Enter "admin" for both the user name and password.
3. Under "Domain: dcg.manager", click **dcg.manager:server=PerformanceMonitorManager**.
4. Click the **Invoke** button next to **enable**.
5. Click **Return to MBean View** to go back to the PerformanceMonitorManager screen.

Performance monitoring is now enabled. When a dataflow runs, the performance statistics will display at the top of the PerformanceMonitorManager screen.



Note the following:

- The statistics are reported in a semicolon-delimited format. The first row is the column header. We recommend putting the data into a spreadsheet for easier viewing.
- The time values in the report (Avg, Min, Max, Total) are displayed in milliseconds.
- You must refresh the screen to see updates.
- To reset the counters, click the **Invoke** button next to **reset**.
- If you stop the Spectrum™ Technology Platform server, performance monitoring will be turned off. You will have to turn it back on when you start the server again.

Monitoring Memory Usage

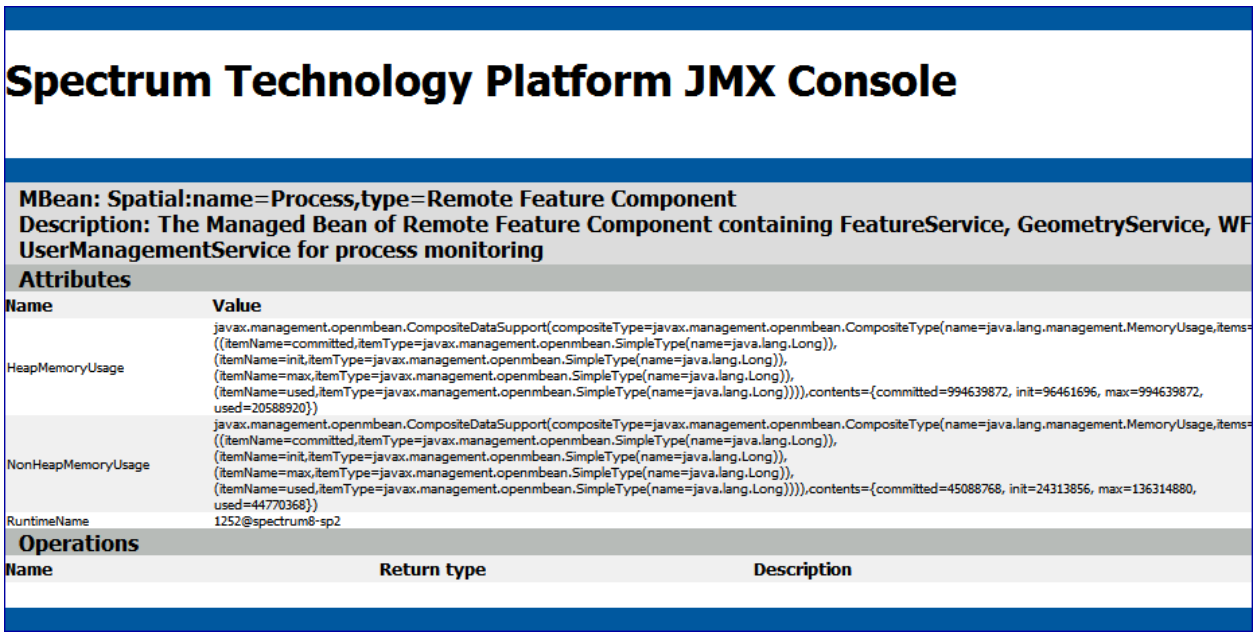
The JMX Console allows you to monitor the JVM heap usage of each remote component.

The monitoring processes for Spectrum Spatial are:

- Spatial:name=Process,type=Remote Feature Component
  - Feature Service
  - Geometry Service
  - Named Resource Service

- User Management Service
- WFS
- CSW

- Spatial:name=Process,type=Remote Mapping Component
  - Mapping Service
  - Map Tiling Service
  - WMS



Spectrum Technology Platform JMX Console

MBean: Spatial:name=Process,type=Remote Feature Component
Description: The Managed Bean of Remote Feature Component containing FeatureService, GeometryService, WF
UserManagementService for process monitoring

Attributes

Name	Value
HeapMemoryUsage	javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.openmbean.CompositeType(name=java.lang.management.MemoryUsage,items=((itemName=committed,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=init,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=max,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=used,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)))),contents={committed=994639872, init=96461696, max=994639872, used=20588920})
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.openmbean.CompositeType(name=java.lang.management.MemoryUsage,items=((itemName=committed,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=init,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=max,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=used,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)))),contents={committed=45088768, init=24313856, max=136314880, used=44770368})
RuntimeName	1252@spectrum8-sp2

Operations

Name	Return type	Description

Memory usage (HeapMemoryUsage and NonHeapMemoryUsage) is based on the standard JVM memory MBean. It shows the memory usage of the JVM that the remote component running on. It includes the amount of init, max, committed and used memory.

RuntimeName includes the process ID that you can use to find more information from the operating system (for example, by using the Windows Task Manager), or even kill the process.

In the heap sections, `={committed=72351744, init=65157504, max=954466304, used=6559552})` are shown in bytes. The number is for a particular remote component, which includes multiple services. Each remote component runs in its own JVM, and the JVM only runs this component.

Init is the initial amount JVM allocated (-Xms); max is the one specified by –Xmx. Used is the amount of memory that used by JVM for objects. The relationship is like this: –Xms < committed < -Xmx, and used < committed.

You can modify the heap memory by modifying the -Xm in the java.vmargs file under the spatial folder (<Installed>\Pitney Bowes\Spectrum\server\modules\spatial\java.vmargs). See **Increasing Heap Memory for more instructions.**

4

Managing Memory and Threading

In this section:

- Introduction to Managing Memory and Threading 42
- Spectrum Performance Tuning 42
- Increasing Heap Memory for Spatial Components 43
- Increasing Heap Memory for the Platform 44

Introduction to Managing Memory and Threading

This section describes approaches for improving performance by managing memory and threading, and also relates best practices for optimizing the performance of the Location Intelligence Module. It is intended for experienced administrators.

Spectrum Performance Tuning

Spectrum provides several tuning options to optimize performance of the server. The optimal selection of settings is dependent on the nature of the deployment. To create a well-tuned server environment, it is recommended that performance tests should be executed in the deployed environment to determine optimal settings. This section provides some general guidance on performance tuning.

JVM Tuning

Spectrum is a Java server, and as a result, JVM tuning parameters can be used to optimize performance of remote components. The JVM can be configured through the `/<spectrum root>/server/modules/spatial/java.vmargs` file.

To optimize Spectrum's performance using JVM tuning parameters:

1. Stop the Spectrum server.
2. Open the java.vmargs file in a text editor. Set the maximum memory allocation on the JVM.

   An allocation of 512m for each active CPU core is generally appropriate. Do not exceed the maximum memory available to your operating system and leave a suitable space for the operating system to do its work.
3. Save the file.
4. Restart Spectrum.

Remote Component Configuration

Each spatial service component in Spectrum is deployed into its own JVM instance separate from the Spectrum Platform run time. This ensures the platform is independent of the modules within it and that JVM configuration can be applied per service, allowing flexibility of memory allocation and tuning for performance based on the characteristics of the service.

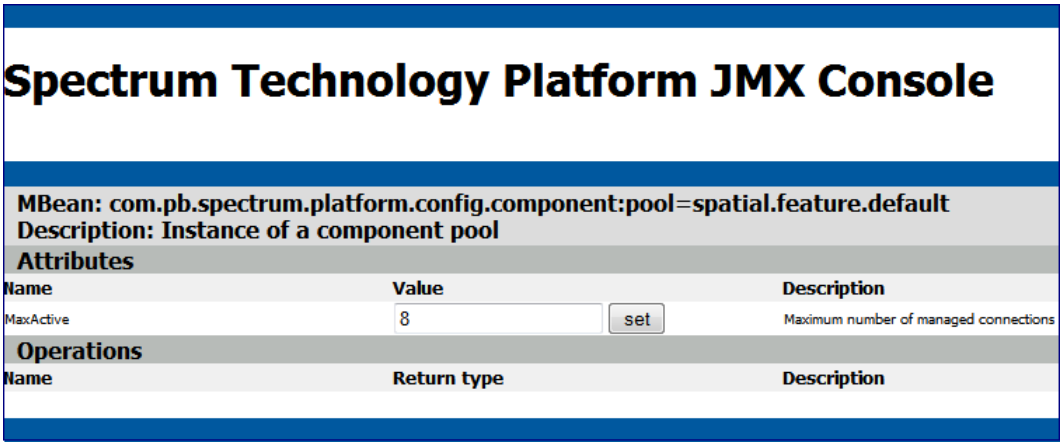
Tuning the Components Using MaxActive

In addition to JVM tuning, components can be tuned using the MaxActive setting. This setting represents the number of threads on the components listening for service requests from the Spectrum Platform (that is, the maximum number of managed connections). Every web service request enters Spectrum from the platform and is passed to the components. This value should be increased to accommodate greater request loads. Performance test should be run with various settings until optimal performance is achieved for the usage. The maximum setting should not go above twice the number of CPU core.

The Spectrum platform provides capability for tuning remote components using the JMX console (`http://host:port/jmx-console/`). The components, each of which includes several services, appear in the Domain: `com.pb.spectrum.platform.config.component` section as follows:

- com.pb.spectrum.platform.config.component:pool=spatial.feature.default
  - Feature Service
  - Geometry Service
  - Named Resource Service
  - User Management Service
  - WFS
  - CSW
- com.pb.spectrum.platform.config.component:pool=spatial.mapping.default
  - Mapping Service
  - Map Tiling Service
  - WMS

A best practice around maxActive threading is to limit the combined number of threads to twice the CPU; for example, on a 4 CPU machine the combined number of threads for all services should not exceed 8.



Increasing Heap Memory for Spatial Components

To increase the heap memory for spatial remote components:

1. Stop the Spectrum server.
2. In a text editor, open the java.vmargs file from `<Installed>\Pitney Bowes\Spectrum\server\modules\spatial\java.vmargs`.
3. Change the vmargs default of 1GB (1024MB). For example, to increase the memory to 2GB, change the vmargs from the default of `-Xmx1024m -Djava.io.tmpdir=../app/tmp` to `-Xmx1536m`

`-Djava.io.tmpdir=../app/tmp`. This increases the memory of each spatial remote component to 1.5GB and enables remote components to connect via Jconsole for debugging.

4. Save the java.vmargs file.
5. Restart the Spectrum server.

Increasing Heap Memory for the Platform

To increase the heap memory for the Spectrum platform:

1. Stop the Spectrum server.
2. In a text editor, open the wrapper.conf file from `<Installed>\Pitney Bowes\Spectrum\server\bin\wrapper`.
3. Change the vmargs default of 1GB (1024MB). For example, to increase the memory to 2GB:

```
# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=2048
```

4. Save the java.vmargs file.
5. Restart the Spectrum server.

5

Load Balancing Spatial Services Tutorial

In this section:

- About this Tutorial 46
- Deployment Architecture 46
- Install Spectrum 48
- Set Up a Map Image File Share 49
- Configure Spectrum 49
- Configure Common Repository 50
- Shared Spectrum Local Data 54
- Performance Tuning 54
- Set Up Load Balancer 54

About this Tutorial

The Spectrum spatial services that are included with the Location Intelligence Module are designed to support enterprise scale deployment requirements. This includes deployment configurations for high availability and support for horizontal and vertical scaling. Spectrum's high availability configuration enables organizations to create fault tolerant deployments supporting requirements for continuous service provision. Spectrum's configurations for horizontal and vertical scaling provide the ability to grow the capacity of a system to support greater load. The horizontal scaling configuration supports additional load through addition of more servers to the cluster. The vertical scaling configuration grows the capacity of the system to support larger loads through the addition of greater hardware resources. Spectrum uses a load balancing based approach to deliver resilience and horizontal scaling architectures.

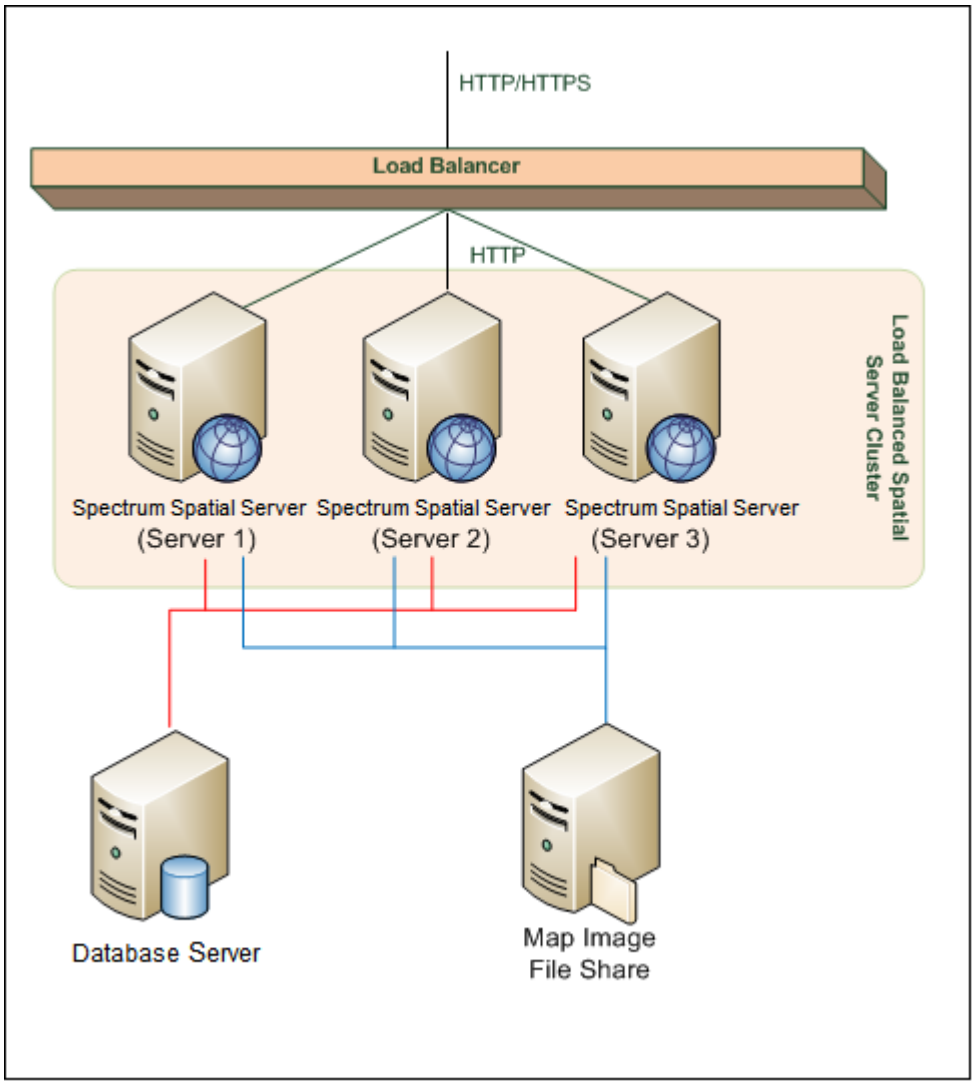
The goal of this tutorial is to illustrate the concepts of load balancing a Spectrum installation with the spatial services for resilience or high capacity. Both the concepts and the steps outlined in this tutorial can apply to virtualized and native environments. The tutorial will provide information on scaling spatial server horizontally and vertically.

This tutorial uses the following hardware/software and Spectrum components:

- Spectrum LIM Mapping Service
- PostGres database 9.0.3
- RedHat Linux 5.4

Deployment Architecture

In this tutorial we will create a load balanced Spectrum deployment. The diagram below illustrates the deployment architecture of the configuration we will create. Load balancing can be used to support high availability and scaling. The deployment architecture includes a load balancer, Spectrum spatial services cluster, database and a file share. With this approach it is possible to scale both horizontally and vertically.



Load Balancer

The load balancer spreads requests between the Spectrum instances. Any load balancer that supports load balancing HTTP/HTTPs requests can be used. Spectrum Cluster The cluster is a collection of Spectrum instances with LIM sharing administration, named resources, geographical metadata content and configuration settings. Additional nodes can be added to the cluster for resilience or to deliver support for greater loads. Each node can be scaled vertically through additional hardware resources and/or additional instances should this be required for hardware with massive resources. Spectrum can be configured to use restricted numbers of CPUs.

Database

Spectrum stores named resources (maps, layers, tables and styles), geographic metadata and configuration in a repository. In the default single server installation an embedded database is used to store these resources on the local server. To create a resilient scalable solution this embedded database should be replaced with a resilient independent database. In this release Spectrum supports Oracle, PostGreSQL(PostGIS) and Microsoft SQL Server as repository databases.

In the load balanced configuration Spectrum nodes cache these resources in a local cache and search index in each node in the cluster. When a Spectrum node receives a request it uses the local cache and index to find resources. Named resources can be added through any node in the cluster. Each node keeps its cache current by checking for differences between its local cache and the central database. This check occurs every 2 seconds by default. Time frequency can be configured. This architecture ensures the server delivers high performance transactions and the load on the repository database is kept to a minimum. If a new Spectrum is added to the cluster the cache and index are created automatically. Such a scenario can occur to remedy a node failure or grow the capability of the deployment.

File Share

The file share provides a folder to hold map images generated by Spectrum. When maps are renderer using the web services the server supports the map images being returned through URLs or returned as a base 64 encoded image. When a URL is returned the map image is stored as a file and served on request of the URL. To ensure any Spectrum node can return the map image a file share is used to store the images.

Install Spectrum

In this step we will create a Spectrum deployment.

To install Spectrum into your VM instance:

1. Copy the Spectrum Linux installer to the target server with Red Hat operating system.
2. Install Spectrum.

   For this tutorial we will follow the default installation but not install the license key. The installation guide for UNIX and Linux provides more information on the installation process.

   a) Locate the install.sh installer
   b) Ensure the user has execute permission

   ```
   chmod a+x install.sh
   ```

   c) Execute install.sh

   ```
   install.sh
   ```

   The installer will walk you through the installation process.
3. Copy the license key into the server/app/import directory of each Spectrum installed.
4. Start Spectrum

   It is possible to install multiple installations of Spectrum on the same operating system. This can be used to provide flexibility when vertically scaling the server. To support multiple installations on the

same machine the hidden file /var/.com.zerog.registry.xml needs to be renamed to enable a new installation to a different folder and port on the same machine.

Spectrum is now installed. The next step is to configure the Spectrum load balanced cluster deployment.

Set Up a Map Image File Share

The file share provides a folder to hold map images generated by Spectrum . Create a shared folder accessible to all Spectrum nodes. The file share is not required if maps are returned from the web services as base 64 encoded images.

To set up a map image file share:

1. Task s Mount a shared folder on each operating system hosting Spectrum . The commands below mount a drive on a Microsoft Windows Server or network drive supporting CIFS.

```
mkdir /mnt/<linux mount>
mount -t cifs //<windows host>/<windows share> /mnt/<linux mount>-o
username=shareuser,password=sharepassword,domain=pbi
```

2. Set the image share to load at startup in `/etc/fstab`.

```
//<windows ip address for share>/share /path_to/mount cifs
username=server_user,password=secret,_netdev 0 0
```

Configure Spectrum

Once Spectrum is installed, you need to configure your instance before you can replicate it to another virtual machine. If you are not using a virtual machine environment, you will need to perform these steps on each of your Spectrum installations.

Add the Map File Share to Spectrum

Context for the current task

1. Modify the Mapping service configuration by pointing to a shared image folder and load balance server. In the ImageCache change the Directory parameter to a common image directory, and change the AccessBaseURL parameter to the load balancer machine image URL.

   If using a virtual machine environment, remember this IP address, as you must set the load balancer VM to this IP in the section **Set Up Load Balancer** on page 54.

```
<ImageCache>
<Directory>/mnt/<linux mount>/images</Directory>
<AccessBaseURL>http://<loadbalance_IP_address>/Spatial/images</AccessBaseURL>

 <FileExpire>30</FileExpire>
 <ScanInterval>30</ScanInterval>
</ImageCache>
```

2. Set up symbolic link to enable map images to go to the shared file system.

```
cd /<spatial server root>/server/modules/spatial
rm –Rf images
ln -s / mnt/<linux mount>/images
```

Modify the Service Configurations

To modify the service configurations for load balancing:

In each service configuration file, change the <RepositoryURL> to point to the load balance server repository URL.

The RepositoryURL should change to point to the balancer from `http://<Spectrum>/RepositoryService/rmi` to `http://<Balancer>/RepositoryService/rmi`.

Modify Java Properties File

To modify the java properties for Spectrum™ Technology Platform:

1. Modify the java.properties file, located in <spectrum>/server/modules/spatial/java.properties, to point to the load balance server.
2. Change the images.webapp.url and all of the service host and port numbers to point to the load balance server.

Configure Ports for Multiple Spectrum Instances

If you have multiple Spectrum™ Technology Platform instances on a single machine, you must change the port numbers.

To change the port numbers for each Spectrum instance:

1. Change all ports in <Spectrum root>/server/app/conf/spectrum-container.properties to new port values that are not in use. The http port reflects the port number entered in the installer.
2. Update the rmi port in bootstrap.properties in the bin/jackrabbit folder (e.g. 11099). The default is 1099.

Configure Common Repository

The next step is to configure Spectrum to use a common repository database for the cluster. This ensures that named resources, geographic metadata and configuration settings are managed across the cluster. The description below describes how to configure Spectrum to use a common repository database. This tutorial will use a PostgreSQL database. It is possible to use an Oracle or Microsoft SQL Server database.

The repository is installed with a set of named resources, geographic metadata and configuration files. To migrate these resources to the common database repository the resources need to be exported from the default internal repository database and reimported into the new shared repository database.

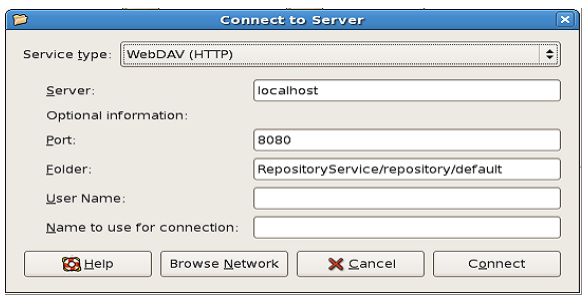
To provide support for bulk export/import of repository content Spectrum repository provides a WebDAV interface.

Bulk Export Using WebDAV

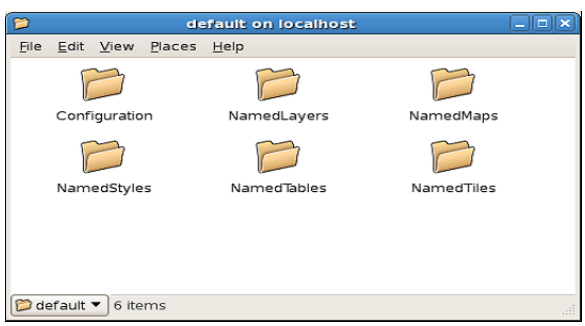
You need to export the contents of the installed repository. This step only needs to be performed once as the contents of the repository should be the same at this point for all instances of Spectrum. There are many free WebDAV clients including Microsoft Windows Explorer and GNOME- Nautilus. For this Linux focused tutorial we will use GNOME- Nautilus to export the resources from Spectrum.

1. Start Spectrum.
2. Connect to Spectrum using WebDAV:
   a) Connect to the WebDAV Directory using GNOME – Nautilus.
   b) Select **Connect to Server** in the Places menu. This will open a File Browser window.

   The repository is at the following location.
   `http://<YourMachine>:<Port>/RepositoryService/repository/default`



3. Copy the content of the repository to a local drive.



Set Up the Common Repository Database

These steps need to be performed on all instances of Spectrum in your load balanced environment:

1. Stop Spectrum.
2. Add the database JDBC drivers to the Spectrum common lib directory to allow it to use the selected database.

In this tutorial we are using PostGreSQL JDBC drivers. Copy the /<spectrum root>/server/modules/spatial/lib/postgresql-8.4-701.jdbc4.jar file to /<spectrum root>/server/app/lib/postgresql-8.4-701.jdbc4.jar

3. Edit the /<spectrum root>/server/modules/spatial/jackrabbit/repository.xml file to point the repository to a database and add clustering. There are four separate changes you need to make:
   a) Modify the two FileSystem sections within the Repository and Workspace sections of the file:

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
   <param name="driver" value="org.postgresql.Driver"/>
   <param name="url"
value="jdbc:postgresql://<hostname>:<port>/<databasename>"/>
   <param name="schema" value="postgresql"/>
   <param name="user" value="<user>"/>
   <param name="password" value="<pwd>"/>
   <param name="schemaObjectPrefix" value="rep_"/>
  </FileSystem>
```

   b) Modify the Persistence Manager within the Workspace:

```
<PersistenceManager
class="org.apache.jackrabbit.core.persistence.bundle.PostgreSQLPersistenceManager">

   <param name="url"
value="jdbc:postgresql://<hostname>:<port>/<databasename>"/>
   <param name="schema" value="postgresql"/>
   <param name="user" value="<user>"/>
   <param name="password" value="<pwd>"/>
   <param name="schemaObjectPrefix" value="${wsp.name}_"/>
   <param name="externalBLOBs" value="false"/>
  </PersistenceManager>
```

   c) Enable Clustering at the end of the file, right above the </Repository> tag. Each instance of Spectrum will need to have a distinct id to enable synchronization of clustering to work. The delay defines the time delay for synchronization in milliseconds.

```
<Cluster id="node1" syncDelay="2000">
   <Journal class="org.apache.jackrabbit.core.journal.DatabaseJournal">

    <param name="revision" value="${rep.home}/revision.log" />
    <param name="driver" value="org.postgresql.Driver" />
    <param name="url"
value="jdbc:postgresql://<hostname>:<port>/<databasename>" />
    <param name="schema" value="postgresql"/>
    <param name="schemaObjectPrefix" value="rep_"/>
    <param name="user" value="<user>"/>
    <param name="password" value="<pwd>"/>
    <param name="databaseType" value="postgresql"/>
   </Journal>
  </Cluster>
```

   d) Comment out the DataStore section:

```
<DataStore class="org.apache.jackrabbit.core.data.FileDataStore"/>
```

4. Remove the following folders from the /server/modules/spatial/jackrabbit directory for each instance of Spectrum: repository, version, workspaces.

5. If your PostGIS database has previously had repository content added, you must remove tables from your database so a clean repository can be created. If you are starting with a new database, please make sure the tables do not exist. The following tables need to be removed from the database:

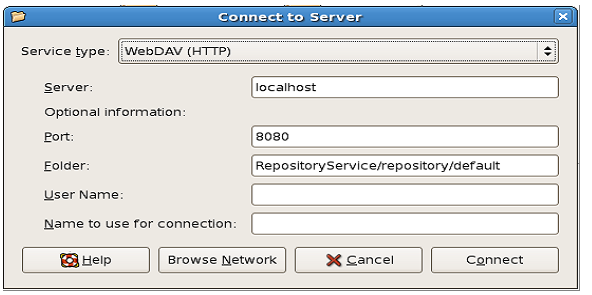
```
public.default_names_id_seq
public.default_binval
public.default_bundle
public.default_names
public.default_refs
public rep_fsentry
public.rep_global_revision
public.rep_journal
public.rep_local_revisions
public.security_binval
public.security_bundle
public.security_names
public.security_refs
```

Import the Repository Content

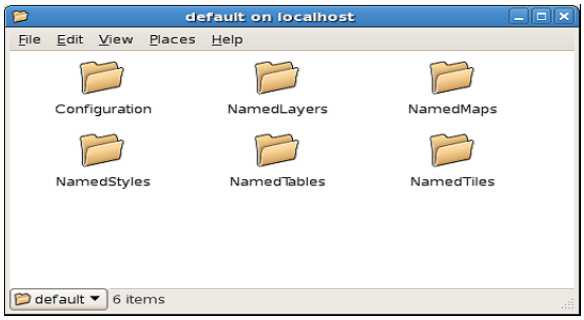
Next, import the content of the repository we previously exported back into the repository. This step only needs to be performed on one of the Spectrum instances.

1. Start Spectrum
2. Copy the previously exported content of the repository back into the repository. First, connect to Spectrum using WebDAV:
   a) Connect to the WebDAV Directory using GNOME – Nautilus. Select “Connect to Server....” in the Places menu. This will open a File Browser window.

      The repository is at the following location.
      `http://<YourMachine>:<Port>/RepositoryService/repository/default`.



   b) Copy the content to the repository root directory.



Spectrum is now configured and ready to be load balanced.

Shared Spectrum Local Data

If you are using TAB file data on the file system, this data needs to be in a shared location accessible by all instances of Spectrum in the load balanced environment. It is also important to note that all named resources in the repository accessing data on the file system should point to this shared location.

Each VM or machine hosting Spectrum needs to have access to the mounted shared drive.

Note: Using named resources that point to database tables do not require a shared drive, as the named resources in the repository do not access the data using a file path; rather they use a named connection to the data in the database.

Performance Tuning

See **Spectrum Performance Tuning** on page 42 for general guidance on optimizing the performance of the Spectrum server.

Set Up Load Balancer

Now that multiple instances of Spectrum are deployed and configured, the instances need to be load balanced. In this tutorial Apache HTTP Server is used as a load balancer. Any load balancer with support for load balancing http requests can be used.

In the Apache HTTP Server configuration (httpd.conf file e.g., `/etc/httpd/conf/`) turn on the proxy and load balance modules by adding the following sections at the end of the file. You will need to add the IP addresses for the number of Spectrum instances you have created (replace `<VMCloneIP1>:<Port Number>` with the correct IP address for each VM or system address.

```
<IfModule proxy_module>
ProxyRequests Off
```

```
<Proxy *>
Order deny,allow
Allow from all
</Proxy>
ProxyPreserveHost On

#Load Balancer Manager
 <Location /balancer-manager>
  SetHandler balancer-manager
Order deny,allow
   Allow from all
  </Location>

 #MappingService
  <Proxy balancer://Spatial >
   BalancerMember http://<VMCloneIP1>:8080/Spatial
   BalancerMember http://<VMCloneIP2>:8080/Spatial
   BalancerMember http://<VMCloneIP3>:8080/Spatial
   BalancerMember http://<VMCloneIP4>:8080/Spatial
  </Proxy>
 ProxyPass /Spatial balancer://Spatial

 #MappingService
  <Proxy balancer://soap>
   BalancerMember http://<VMCloneIP1>:8080/soap
   BalancerMember http://<VMCloneIP2>:8080/soap
   BalancerMember http://<VMCloneIP3>:8080/soap
   BalancerMember http://<VMCloneIP4>:8080/soap
  </Proxy>
 ProxyPass /soap balancer://soap

#RepositoryService
<Proxy balancer://RepositoryService>
                  BalancerMember http://<VMCloneIP1>:8080/RepositoryService
                  BalancerMember http://<VMCloneIP2>:8080/RepositoryService
                  BalancerMember http://<VMCloneIP3>:8080/RepositoryService
                  BalancerMember http://<VMCloneIP4>:8080/RepositoryService
</Proxy>
      ProxyPass /RepositoryService balancer://RepositoryService

#REST Service
<Proxy balancer://rest>
      BalancerMember http://<VMCloneIP1>:8080/rest
      BalancerMember http://<VMCloneIP2>:8080/rest
      BalancerMember http://<VMCloneIP3>:8080/rest
      BalancerMember http://<VMCloneIP4>:8080/rest
</Proxy>
      ProxyPass /rest balancer://rest
</IfModule>
```

For load balancing Spectrum, the following modules need to be loaded.

- LoadModule proxy_module modules/mod_proxy.so
- LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
- LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
- LoadModule proxy_http_module modules/mod_proxy_http.so
- LoadModule proxy_connect_module modules/mod_proxy_connect.so

You do not need to perform this task for load balancing the Apache HTTP Server that ships with the CentOS. These modules are loaded by default.

Once you have updated your conf file save the changes and restart the load balancer. Use the following command to restart the Apache HTTP Server load balancer:

- service httpd start
- chkconfig httpd on

Note: Sending a request through the load balance server might time out if the firewall of the Spectrum instances are turned on. In this scenario, either turn off the firewall on these machines, or modify the service configuration files (e.g., MappingConfiguration) to point to the Spectrum instance for that machine. For instance change the RepositoryURL from the load balance server `http://<Balancer>/RepositoryService/rmi` to the local Spectrum instance `http://<Spectrum>/RepositoryService/rmi`.

6

Troubleshooting Your System

In this section:

- Rebuilding a Corrupt Repository Index 58
- Monitoring Memory Usage of a Non-Responsive Server .58

Rebuilding a Corrupt Repository Index

Sometimes the repository can become corrupt if the server is shut down abruptly or the Java process is killed (manually or due to a power outage). As a result, you may be unable to get resources that were previously searchable, and there will be no errors or warnings in the logs. Once you verify that permission changes are not the cause, rebuild the index to fix this issue:

1. Shut down the server.
2. Delete the index directory at the following locations:
   - <Spectrum>\server\modules\spatial\jackrabbit\workspaces\default
   - <Spectrum>\server\modules\spatial\jackrabbit\workspaces\security
   - <Spectrum>\server\modules\spatial\jackrabbit\repository
3. Restart the server.
   Jackrabbit re-creates the index at the above locations while booting.

After rebuilding the index, the search works correctly again.

Monitoring Memory Usage of a Non-Responsive Server

If your Spectrum server stops responding, you can follow the steps below to monitor its performance and resource consumption. This monitoring provides information you can use to adjust memory and threading usage.

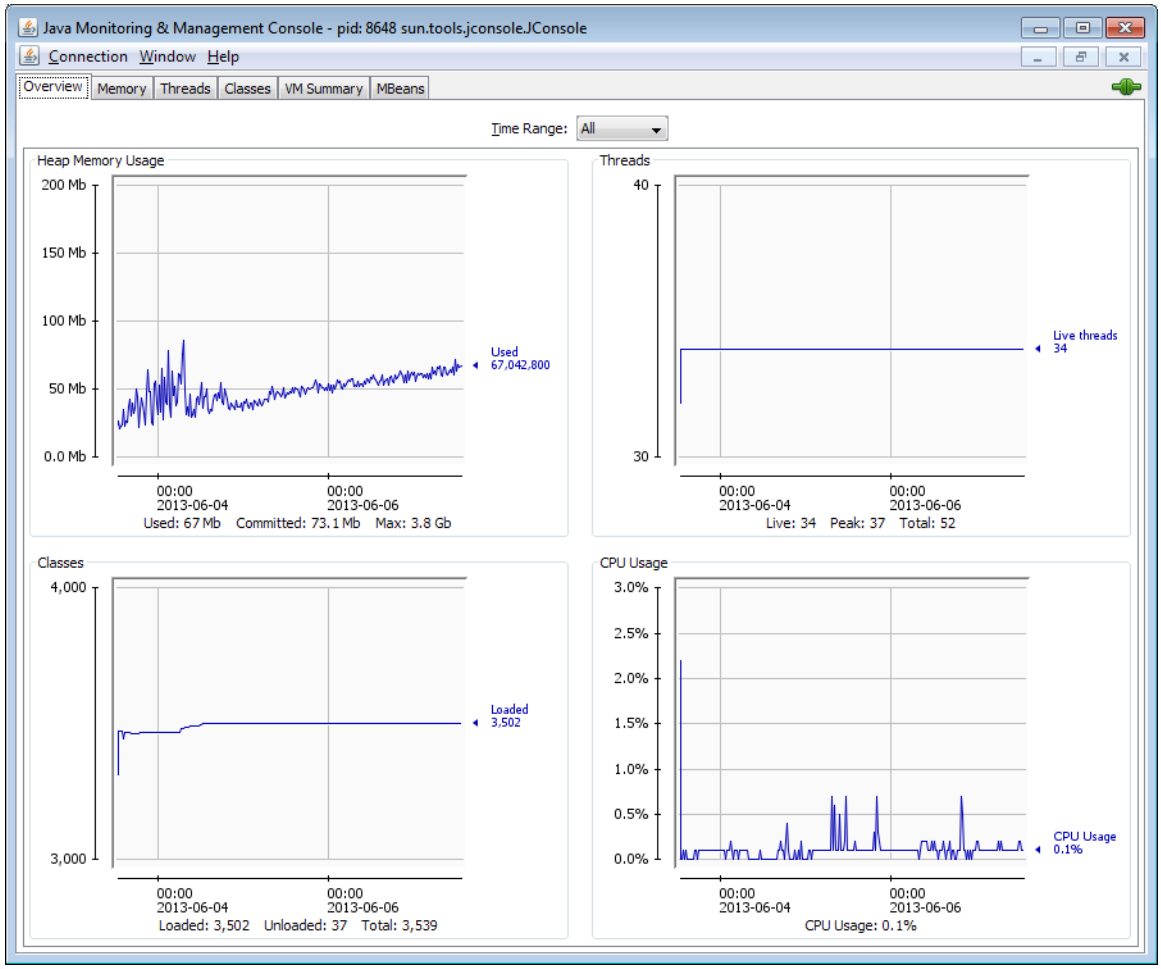
1. Check whether a service other than the Mapping Service is working. For example, start the Feature Service on the demo page: http://<servername>:<port>/Spatial/FeatureService//DemoPage.html. This determines whether the whole server is down or just the Mapping Service.
2. Verify you have enough disk space for both Mapping and MapTiling images to be stored by inspecting the configuration files:
   - **Mapping:**
     http://localhost:8080/RepositoryService/repository/default/Configuration/MappingConfiguration under "`<Directory> C:\Program Files\Pitney Bowes\Spectrum/server/modules/spatial/images </Directory>`"
   - **MapTiling:**
     "http://localhost:8080/RepositoryService/repository/default/Configuration/MapTilingConfiguration" under "`<Property name="diskPath" value="C:/Program Files/Pitney Bowes/Spectrum/server/modules/spatial/TileCache"/>`"
3. Stop the Spectrum server.
4. In a text editor, open the java.vmargs files from `<Installed>\Pitney Bowes\Spectrum\server\modules\spatial\java.vmargs`.

5. Change the vmargs from the default of `-Xmx1024m -Djava.io.tmpdir=../app/tmp` to `-Xmx1536m -Djava.io.tmpdir=../app/tmp` . This increases the memory of each spatial remote component to 1.5GB and enables remote components to connect via Jconsole for debugging.

   Note: You can increase memory to 2 GB if you have enough memory on the server (for example,-Xmx2048m).

6. Save the java.vmargs file.
7. Start the server wrapper:
   a) Open a command prompt as Administrator.
   b) Go to `<Installed>\Pitney Bowes\Spectrum\server\bin\wrapper` directory and type **wrapper.exe -c**.

   This Spectrum server will start in a few minutes.
8. When the server is started, run the following requests from the demo pages:
   a) Open http://<servername>:<port>/Spatial/MappingService/DemoPage.html and run the List Named Maps request.
   b) Open http://<servername>:<port>/Spatial/FeatureService/DemoPage.html and run the List Named Tables request.
9. Go to <Installed>\Pitney Bowes\Spectrum\java64\bin and run jconsole.exe.
10. Under Local Process, select the wrapper process.
11. In Jconsole, add a new session and select the Feature Service process.
12. In Jconsole, add a new session and select the Mapping Service process.
13. Leave Jconsole running to monitor the memory, CPU, threads, and so on for the Spectrum Platform wrapper for Feature Service and Mapping Service.


Java Monitoring & Management Console - pid: 8648 sun.tools.jconsole.JConsole
Connection Window Help
Overview Memory Threads Classes VM Summary MBeans
Time Range: All
Heap Memory Usage
200 Mb
150 Mb
100 Mb
50 Mb
0.0 Mb
Used
67,042,800
00:00
2013-06-04
00:00
2013-06-06
Used: 67 Mb Committed: 73.1 Mb Max: 3.8 Gb
Threads
40
30
Live threads
34
00:00
2013-06-04
00:00
2013-06-06
Live: 34 Peak: 37 Total: 52
Classes
4,000
3,000
Loaded
3,502
00:00
2013-06-04
00:00
2013-06-06
Loaded: 3,502 Unloaded: 37 Total: 3,539
CPU Usage
3.0%
2.5%
2.0%
1.5%
1.0%
0.5%
0.0%
CPU Usage
0.1%
00:00
2013-06-04
00:00
2013-06-06
CPU Usage: 0.1%

Index

D

DAVExplorer
- accessing resources with 16
- adding resources with 16, 17

Debug logging level 34

E

Error logging level 34
Event Log node 34
Event Logging options 34
Events subnode 34

F

Fatal logging level 34

I

Info logging level 34

L

License Information Subnode 38

N

Named Resource Service
- about 15

named resources
- accessing 15–17
- accessing with DAVExplorer 16
- accessing with WebFolders 15
- adding with DAVExplorer 16, 17
- adding with WebFolders 15
- sample files 15
- uploading 15–17

Notification subnode 37

R

repository
- accessing named resources 15–17
- accessing resources with DAVExplorer 16
- accessing resources with WebFolders 15
- adding resources with DAVExplorer 16, 17
- adding resources with WebFolders 15
- uploading named resources 15–17

S

services
- configuring 17–18

T

Trace logging level 34

U

users
- adding new 24
- deleting 25
- modifying 25

V

Version Information Subnode 38

W

Warn logging level 34
web services
- configuring 17–18

WebDAV protocol tool
- about 15

WebFolders
- accessing resources with 15

WebFolders *(continued)*
adding resources with 15